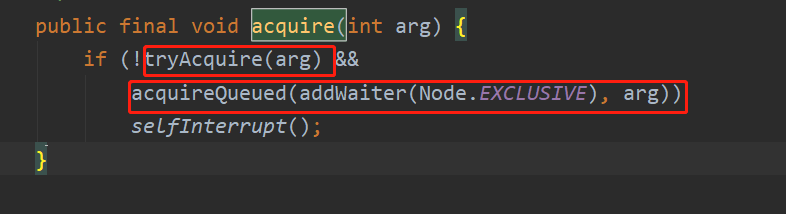




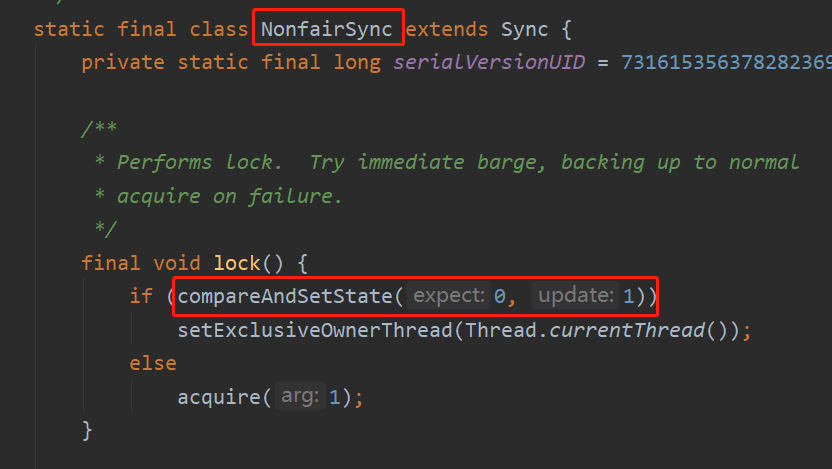
公平锁acquire（lock）时需要判断是否有向前等待队列

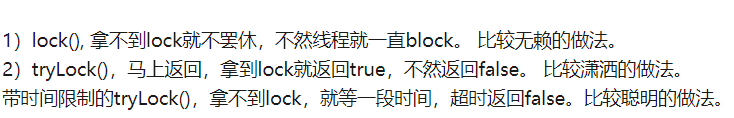
公平锁

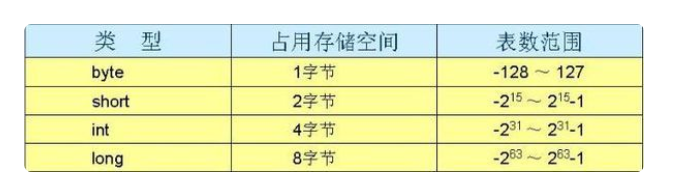
获取lock失败，进入等待队列



非公平锁一进来就去竞争锁了







POST departgrouppersoninfo/\_update/1301702073619841024

{

"doc": {

"fee": {"isPaid" : "0",

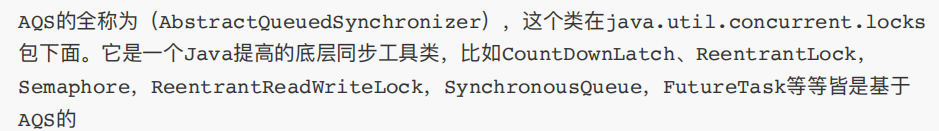
"isInsurance" : "1",

"isDeposit" : "1"}

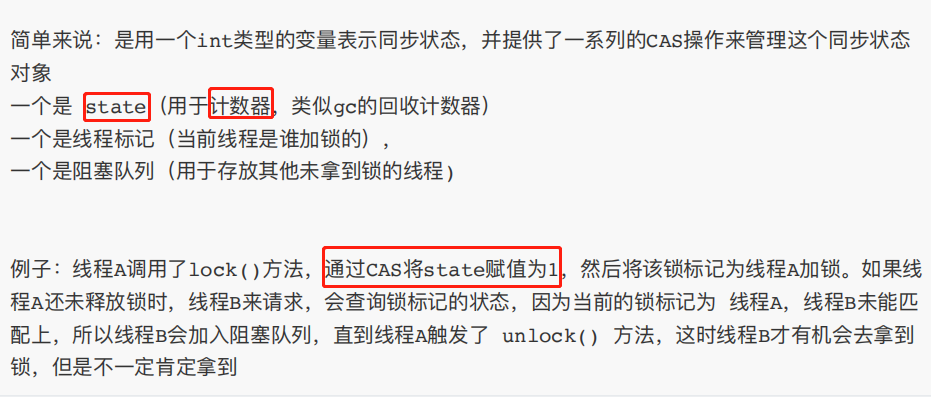
}

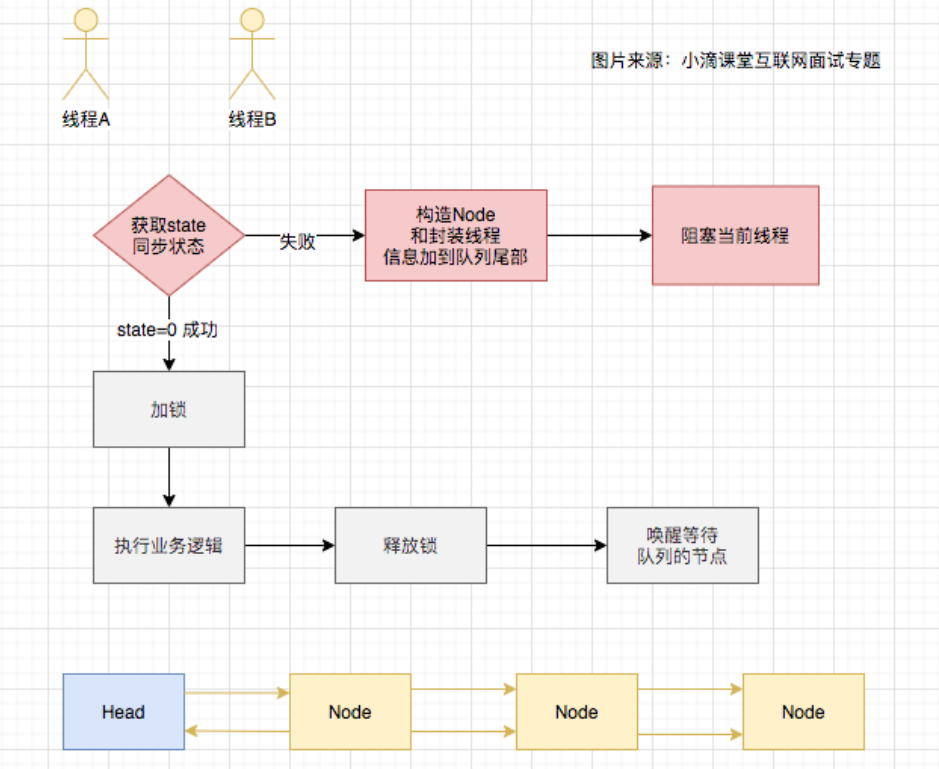
}

GET departgrouppersoninfo/\_doc/1301713871186165760



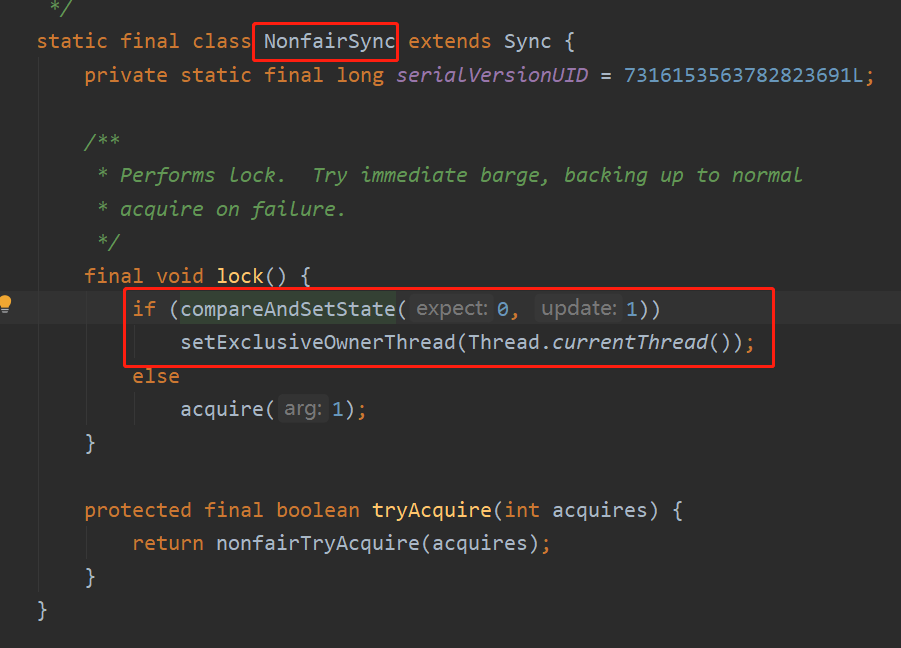


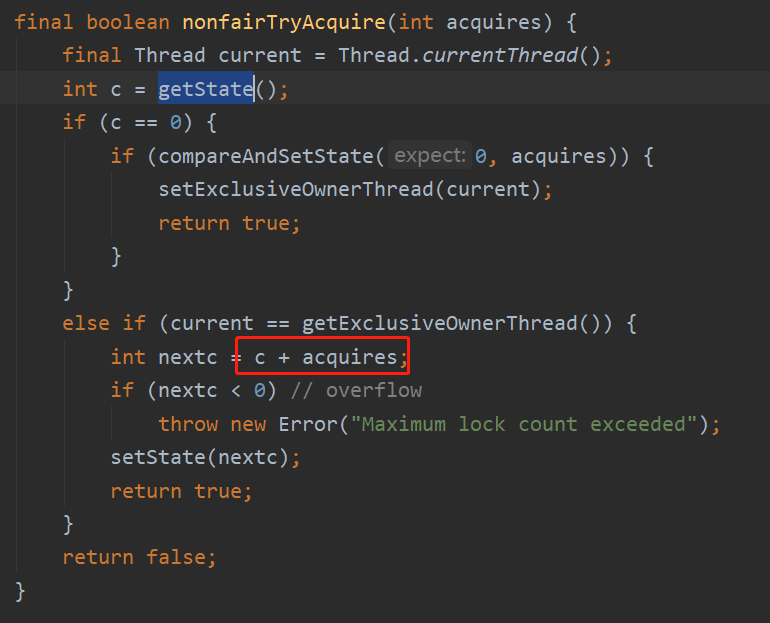




阻塞队列是一个双向队列

ReentrantLock非公平锁一进来就cas比较是否state初始值是0

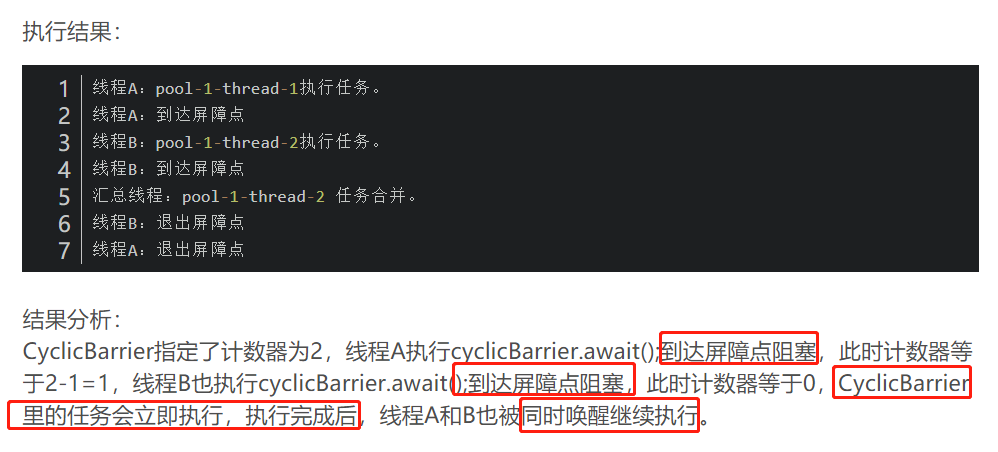




重入锁到达2的31次方-1就变成负数

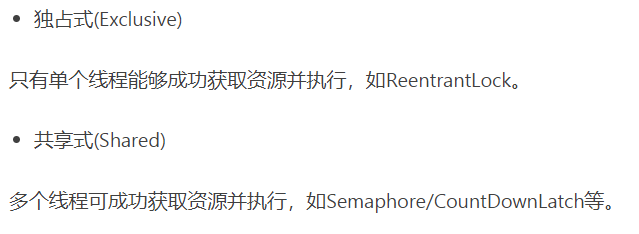
<https://blog.csdn.net/fu123123fu/article/details/103731228>

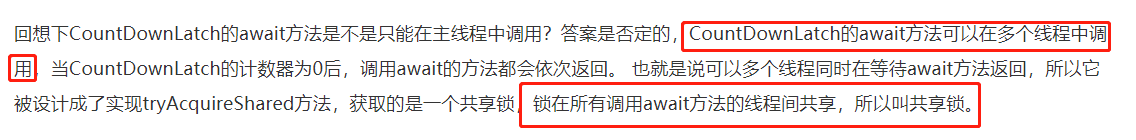


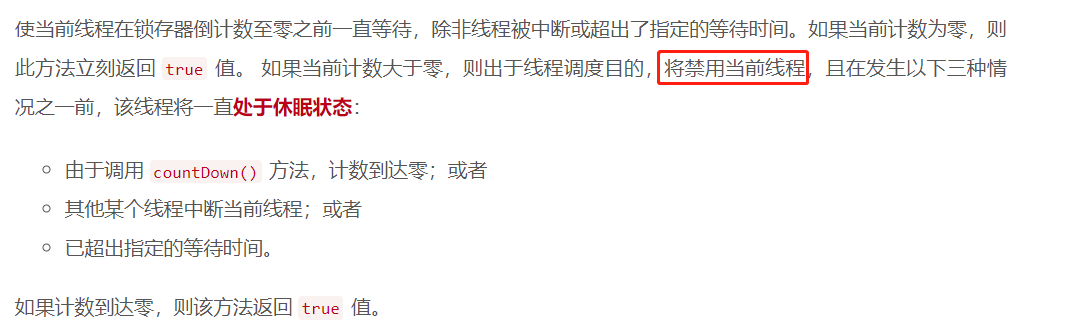


AQS

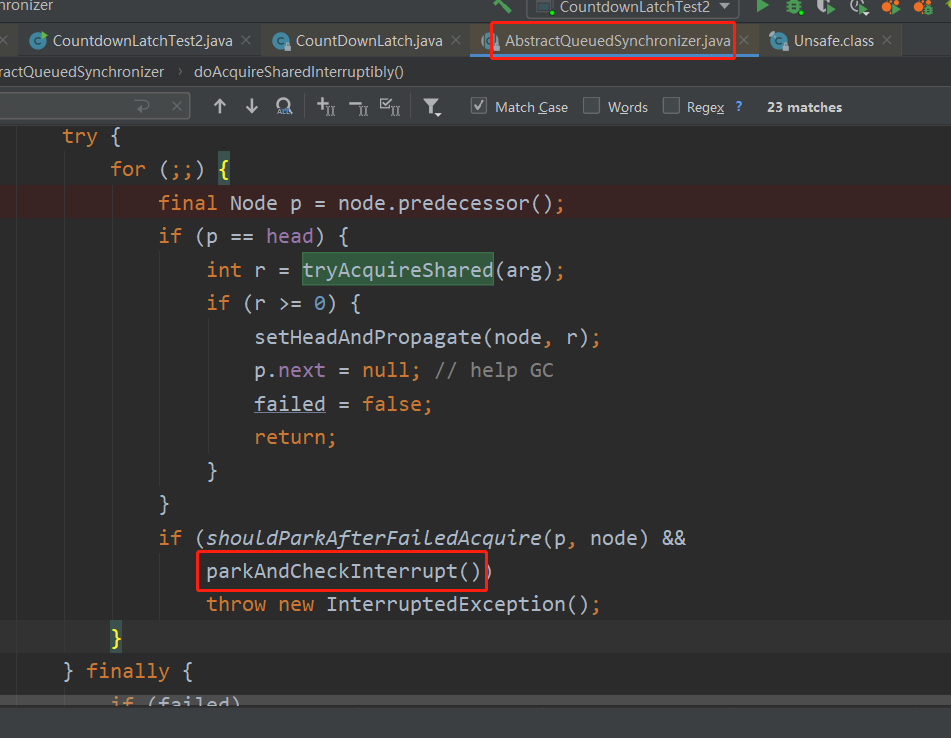


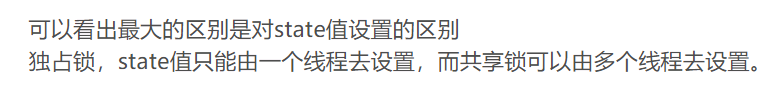




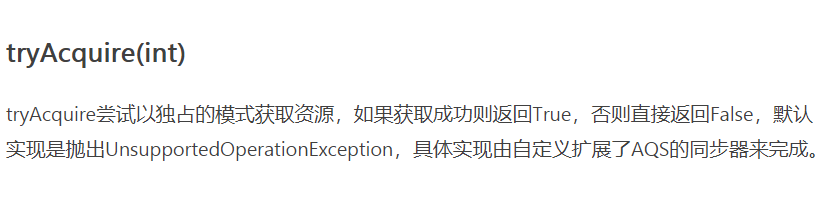


CountDownLatch中await获取锁失败后悔挂起

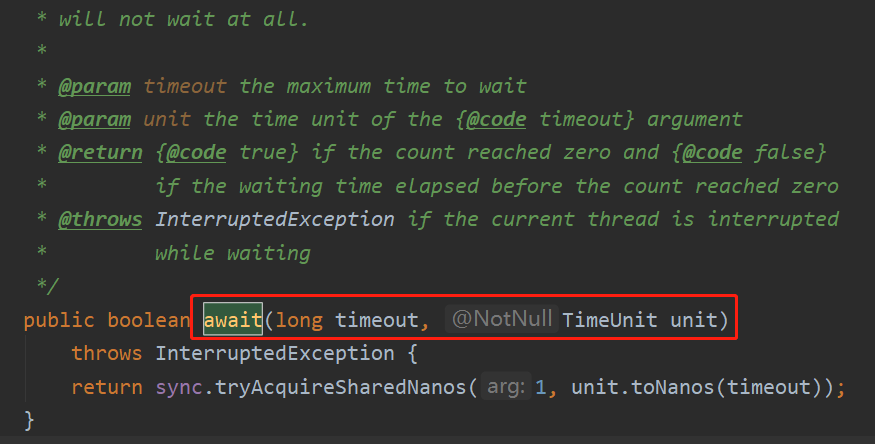




非公平独占锁，A线程获取锁，判断state是否为0，为0则获取到锁，此时另一个线程进来他会判断state是否为0，不为0的话判断是否是当前锁的可重入锁，是的话state加一，但是到达2的31次方-1的话会触发异常超过最大可重入锁数量。如果状态不为1也不是可重入锁，就进入等待队列，等待当前状态为0重新获取锁。而共享锁，state可以自己设置，比如new一个countdownlatch,此时设置state为5，可以5个线程同时改变这个state，从而把它降为0触发await







CoundownLatch的await可以设置超时时间